СОДЕРЖАНИЕ

Введение..... 3
1 Аналитический обзор существующих методов и средств автоматизации 4
1.1 Анализ предметной области..... 4
1.2 Аналитический обзор существующих аналогов..... 5
1.3 Обзор технологий для реализации программного комплекса..... 9
1.4 Постановка задачи на дипломное проектирование..... 13
2 Архитектура программного обеспечения..... 15
2.1 Функциональное моделирование..... 15
2.2 Разработка информационной модели..... 17
2.3 Проектирование базы данных..... 19
Заключение..... 25
Список использованных источников..... 26

ВВЕДЕНИЕ

Расписание занятий в высшем учебном заведении служит для сведения в единую взаимосвязанную систему обучающихся, преподавателей, учебных предметов и назначенных для проведения занятий мест – аудиторий. Оптимизация расписания занятий является одним из основных факторов, способных существенно оптимизировать учебный процесс[1].

На сегодняшний день проблемным остается вопрос о составлении учебного расписания, так как это весьма затратный процесс и в плане времени, и в плане соблюдения всех требований министерства образования[1].

Многие вузы до сих пор используют ручной режим составления расписания: предполагается минимум два методиста на факультет для составления расписания очной и заочной форм обучения. Часто бывает так, что из-за человеческого фактора появляются нестыковки и накладки в аудиторном фонде и между общеуниверситетскими преподавателями. Как правило, такое расписание составляется на листах со сводной сеткой по дням недели, каждая дисциплина и преподаватель вписываются мелким почерком, а далее распространяется по всем кафедрам и факультетам, где каждый преподаватель должен делать для себя выписку с расписанием проведения пар. Усложняется ситуация тем, что многие университеты имеют несколько учебных корпусов, поэтому необходимо учитывать время перемещения студентов и преподавателей между ними. Такие сложности наравне с трудозатратами являются предпосылками к автоматизации ввода и хранения данных. Предлагаемые для учебных заведений программные продукты позволяют оптимально формировать расписание в автоматическом, а не ручном режиме, буквально несколькими операциями[1].

В основе составления расписания занятий лежит теория расписания. Теория расписания является хорошо изученной и описанной во многих работах, начиная с 1960-х годов. Она широко используется как при организации работы предприятий, так и применима для учебных заведений[2][3][4][5].

Целью написания данной работы является разработка программного комплекса управления расписанием в ГГТУ им. П. О. Сухого, позволяющего формировать расписание в автоматическом режиме. А вот какие задачи являются обязательными для достижения цели диплома:

– выполнить обзор и анализ информации по предметной области и по разработке автоматизированных систем управления расписанием, выполнить анализ существующих технологий и систем программирования, систем управления базами данных;

– разработать функциональную модель системы управления расписанием;

– разработать структуру и состав информационного обеспечения для сопровождения программного комплекса.

1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ АВТОМАТИЗАЦИИ

1.1 **Анализ предметной области**

1.1.1 Расписание занятий с точки зрения формализации в теории расписаний есть определение на шкале времени места проведения занятий по заданным дисциплинам обучения с выполнением предъявляемых к ним требованиям. Требования формируются участниками учебного процесса и руководящими документами[2].

Исходными составляющими данного процесса являются:

– **P** – потоки обучения, которые включают в свой состав от одной до нескольких групп обучения, или подгруппы, которые образуются за счет деления одной или нескольких групп (поток) на отдельные подразделения;

– **T** – преподаватели, являющиеся основным механизмом воздействия на потоки обучения. В отличие от классических подходов в теории расписания (один механизм - одна операция) в данной ситуации могут быть ситуации, когда несколько преподавателей могут объединяться в один «механизм» для проведения занятия;

– **D** – дисциплины обучения, основой которой является тематический план обучения, включающий различные типы занятий;

– **A** – аудитории, являющиеся местом для проведения занятий (выполнения операции). Во многих существующих теориях и системах, реализующих составление расписания, данный участник выносится из общей задачи с целью упрощения системы.

1.1.2 Постановка задачи на планирование.

Во многих работах она формулируется, как перебор всевозможных вариантов для всех исходных данных процесса:

$$R = \{P * T * D * A * z\}, \tag{1.1}$$

где z – периоды проведения занятия (дата и пара).

При таком подходе делается вывод о сложности составления расписания, так как при ее решении появляется экспоненциальный рост количества сочетаний, что делает задачу NP-полной[2][3].

Однако такой подход является не всегда корректным, так как уже на предварительной подготовке к планированию данное сочетание сокращается за счет объединения преподавателя, потока, аудитории (или возможной аудитории) и проводимого занятия по тематическому плану в одну единицу планирования – называемого часто занятие.

Если рассматривать планирование в данном случае как процесс определения временного отрезка для конкретного занятия, то задача становится классической с точки зрения теории расписаний. То есть, для заданного числа

работ (дисциплин) и операций (занятие дисциплины) определить такие временные отрезки, чтобы составленное расписание соответствовало заданным критериям оптимальности и предъявляемым требованиям.

При этом для составления расписания должны быть определены:

– дисциплины и занятия по ним (работы и операции) – основой является тематический план изучения дисциплины;

– преподаватели и аудитории (машины и место расположения машин) для проведения занятий. В большинстве случаев по всем занятиям в тематическом плане определены преподаватели (жесткая привязка) и возможные аудитории (плавающая привязка);

– порядок прохождения занятий (операций по машинам). В настоящее время данный пункт при составлении расписания занятий опускается на основе допустимого предположения, что при усвоении материалов дисциплины он не является критичным. То есть с точки зрения составления расписания порядок проведения занятий будет являться случайным;

– критерий оценки расписания – некоторый параметр, вычисляемый по полученному расписанию, показывающий его оптимальность с заданных точек зрения. Для учебного процесса критерием оценки расписания выступает многопараметрическая функция, включающая как дискретные обязательные требования, так и рекомендуемые оптимизационные требования. Во многих работах данная функция является определяющей с точки зрения составления оптимального расписания. В качестве ее часто используют функцию суммы штрафов, позволяющей достаточно просто оценить оптимальность составленного расписания.

Для автоматизированного составления расписания достаточно задать первые два пункта. Важным моментом для автоматизации является создание такого подхода, который бы позволял оператору делать расписание по выбранным занятиям за минимальное время с контролем на уровне программы или оператором визуально за выполнением заданных требований. Фактически в данном случае решение задачи сводится к поиску наилучшего интерфейса работы программы и является чисто инженерной задачей. Для облегчения работы оператора на первом шаге автоматизации можно использовать алгоритм динамического программирования. То есть определяется порядок составления расписания по дисциплинам таким образом, чтобы разделить процесс планирования на подзадачи, составление расписания по которым будет являться относительно несложным. При этом основной задачей в этом случае будет являться определение критерия ранжирования дисциплин.

Для данного подхода можно использовать следующий критерий[2][8]:

$$K_{opt}(i,j) = W_{rd}(i)/W_{mo}(j), \tag{1.2}$$

где i – номер преподавателя (машины), лежащее в диапазоне от 1 до M;

j – номер дисциплины (работы), лежащее в диапазоне от 1 до N;

W_{rd} – количество занятий (операций), которое необходимо провести по выбранной дисциплине (работе);

W_{mo} – количество занятий (операций), которые может провести преподаватель по планируемому периоду времени с учетом наложенных ограничений на выполнение занятий по данной дисциплине и преподавателю.

Ранжирование дисциплин для составления расписания выполняется в порядке уменьшения полученного критерия.

В теории расписаний данный критерий часто обозначают как резерв времени на выполнение работ – разница между количеством времени машины и количеством времени на работу[5]. В данном случае предлагается использовать отношение этих переменных, позволяющее не только расставить дисциплины по порядку планирования, но и проверить также возможность планирования работ по формуле:

$$K_j = Sum\big(K_{opt}(i,j); N\big) < 1, \tag{1.3}$$

где N – количество дисциплин по данному преподавателю.

Если полученный показатель будет больше 1, то это означает, что преподаватель не располагает достаточным временем для проведения занятий. В этом случае необходимо снизить ограничения на проведение данных занятий, либо выполнить замену преподавателя.

Составленное расписание занятий по данному алгоритму будет являться лишь частично оптимальным. Но в большинстве случаев современных подходов к составлению расписания занятий полученное расписание есть некий компромисс при проведении занятий, полученный на основе опыта операторов и требований, предъявляемых к проведению занятий.

При реализации данного подхода необходимо использовать итерационный метод. То есть при составлении расписания после каждой дисциплины выполняется повторное ранжирование работ.

При возникновении ситуации, когда по выбранной очередной дисциплине планирование является невозможным, выполняется перепланирование. В качестве перепланируемой дисциплины выбирается та, у которой наименьшее значение показателя $K_{opt}(i,j)$.

Для решения проблемы ограничений, накладываемых со стороны составляющих процесса планирования возможно использовать рекуррентный алгоритм адаптации процесса планирования.

1.2 **Аналитический обзор существующих аналогов**

1.2.1 Система «АВТОРасписание» предназначена для быстрого, удобного и качественного составления расписаний занятий и сопровождения их в течение всего учебного года.

Имеется восемь основных модификаций программы для различных учебных заведений: для средних общеобразовательных школ, лицеев и

гимназий; колледжей, техникумов и профессиональных училищ; училищ искусства и культуры; вузов (очная форма обучения); военных вузов; учебных центров, УПК и ИПК; вузов с несколькими удаленными учебными корпусами с учетом времени переезда между ними (очная и заочная формы обучения, сетевая версия).

Интерфейс программы представлен на рисунках 1.1, 1.2.

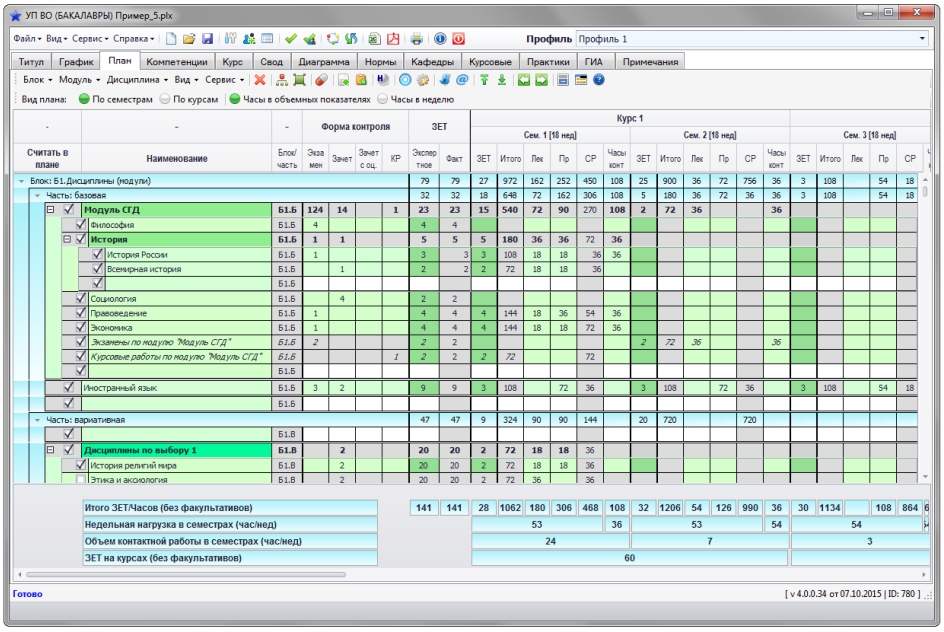

Рисунок 1.1 – Интерфейс программы AVTOR (активная вкладка «План»)

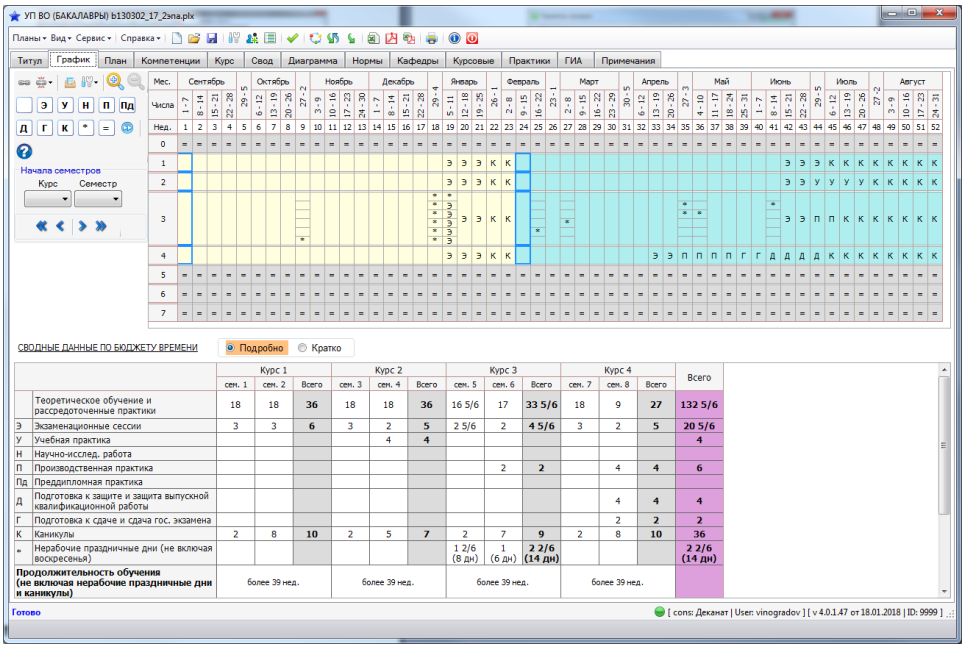

Рисунок 1.2 – Интерфейс программы AVTOR (активная вкладка «График»)

AVTOR позволяет максимально облегчить и автоматизировать сложный труд составителей расписания. Система помогает легко строить, корректировать и распечатывать в виде удобных и наглядных документов:

– расписания занятий классов (учебных групп);

– расписания преподавателей;

– расписания занятости аудиторий (кабинетов);

– учебные нагрузки.

AVTOR-2 имеет приятный дизайн и дружественный сервис. Программа достаточно проста в освоении. Имеется подробное «Руководство пользователя» и справочная система (HELP), где описаны все возможности и способы работы с программой.

Программа отличается уникальным и очень мощным алгоритмом построения и оптимизации расписания. Этот алгоритм является оригинальной авторской разработкой. Он позволяет находить оптимальные решения даже при очень сложных исходных данных.

Полученное автоматическое расписание практически не требует ручной доработки; даже при очень сложных и жестких ограничениях автоматически размещаются все возможные занятия. Если в исходных данных имеются неразрешимые противоречия, то их можно обнаружить и устранить, используя блок анализа.

Технические характеристики: Время работы программы зависит от размерности учебного заведения и мощности компьютера. Полный расчет и оптимизация расписания школы среднего размера со сложными исходными данными (40 классов, 80 преподавателей, из них более 10 совместителей; две смены; дефицит аудиторий) идет около 2-3 минут на компьютере типа Celeron-2000.

AVTOR позволяет:

– строить расписание без «окон» у классов (учебных групп);

– оптимизировать в расписании «окна» преподавателей;

– учитывать требуемый диапазон дней/часов для классов, для преподавателей и для аудиторий;

– учитывать характер работы и пожелания как штатных сотрудников, так и совместителей-почасовиков;

– оптимально размещать занятия по кабинетам (аудиториям) с учетом особенностей классов, предметов, приоритетов преподавателей и вместимости кабинетов;

– вводить расписание звонков;

– устанавливать время перехода (переезда) между учебными корпусами;

– оптимизировать количество переходов из кабинета в кабинет, и из корпуса в корпус;

– легко соединять любые классы (учебные группы) в потоки при проведении любых занятий;

– разделять классы (учебные группы) при проведении занятий по иностранному языку, физической культуре, труду, информатике (и любым другим предметам) на любое количество подгрупп (до десяти!);

– вводить комбинированные уроки для подгрупп (типа «иностранный/информатика») по любым предметам;

– вводить (помимо основных предметов) спецкурсы и факультативы;

– оптимизировать равномерность и трудоемкость расписания;

– легко и быстро вводить и корректировать исходные данные;

– иметь любое количество вариантов расписаний;

– автоматически преобразовывать расписания при изменении базы данных;

– легко сохранять в архивах, копировать и пересылать по E-mail полные базы данных и варианты расписаний;

– быстро вносить любые необходимые корректировки в расписание;

– находить замены временно отсутствующих преподавателей;

– автоматически контролировать расписание, исключая любые «накладки» и противоречия;

– выводить расписания в виде удобных и наглядных документов: текстовых, Word, HTML, а также файлов dBase и книг Excel;

– выставлять готовые расписания в локальной сети и на Интернет-страницах для общего доступа[9].

1.2.2 Система «Электронное расписание» специально разработана с учетом требований учреждений высшего профессионального образования, содержит функции планирования и составления расписания.

Преимущества использования системы:

– мгновенный выбор расписания с любого устройства как для группы, так и для преподавателя. Мгновенное отображение любых изменений;

– при составлении расписания диспетчер сразу видит свободных преподавателей и доступные аудитории, что не допускает случайного наложения занятий;

– успешное внедрение информационных технологий в учебный процесс является важным фактором для повышения имиджа Вашего учебного учреждения[10].

Интерфейс программы «Электронное расписание» отражен на рисунке 1.3.

Рисунок 1.3 – Интерфейс программы «Электронное расписание»

1.2.3 Система «Расписание ПРО» позволяет преподавателям и учителям значительно сократить время составления школьных или институтских расписаний при минимальных затратах усилий.

Основная технология работы представлена в древовидной структуре «Управление» – необходимо пройти последовательно по «веткам» этого дерева сверху вниз. Каждый вариант расписания рассматривается как отдельный проект. Введя однажды исходные данные, можно в дальнейшем делать копию проекта и работать (редактировать и экспериментировать) с ней.

Программа поддерживает два режима управления данными: ручной и автоматический.

Вывести на печать готовое расписание по группам или по преподавателям можно непосредственно из программы, воспользовавшись меню «Файл». Если необходимо дополнительное оформление внешнего вида расписания, можно произвести экспорт результатов в Microsoft Excel.

1.3 **Обзор технологий для реализации программного комплекса**

Для того, чтобы выбрать стек технологий для реализации программного комплекса, необходимо решить, на каких платформах он будет использоваться. Итоговая программа разрабатывается под ОС *Windows* и является *desktop*-приложением.

В качестве языка программирования выбран *Python3.9*.

Python – это язык программирования с открытым исходным кодом, созданный голландским программистом Гвидо ван Россумом и названный в честь британской труппы комиков «Монти Пайтон» (Monty Python). Одним из ключевых соображений ван Россума было то, что программисты тратят больше времени на чтение кода, чем на его написание, поэтому он решил создать легко

читаемый язык. *Python* является одним из самых популярных и простых в освоении языков программирования в мире[11].

Для того, чтобы ускорить разработку, сократить число ошибок и время, затраченное на их исправление, необходимо выбрать удобную и многофункциональную интегрированную среду разработки (IDE). В качестве IDE выбран *PyCharm*. *PyCharm* делает разработку максимально продуктивной благодаря функциям автодополнения и анализа кода, мгновенной подсветке ошибок и быстрым исправлениям. Автоматические рефакторинги помогают эффективно редактировать код, а удобная навигация позволяет мгновенно перемещаться по проекту. [12]

Кроме того, *PyCharm* позволяет установить полезные дополнения. Дополнения обеспечивают:

– подсказки с использование искусственного интеллекта;

– запуск переводчика прямо из кода программы;

– проверка типов переменных в Python (чтобы не допускать ошибки, связанные с типом возвращаемых значений);

– проверка кода программы на уязвимости в автоматическом режиме;

– просмотр файла не открывая его;

– запуск блока кода в консоли без запуска самой программы;

– выделение названий разными цветами (к примеру, чтобы не забыть отредактировать выделенный фрагмент);

– изменение вида редактора в целом;

– и многое другое.

В качестве системы управления базами данных выбрана *PostgreSQL*.

PostgreSQL является одной из самых популярных баз данных. За более чем 20-летнюю историю развития на прочном фундаменте, заложенном академической разработкой, *PostgreSQL* выросла в полноценную СУБД уровня предприятия и составляет реальную альтернативу коммерческим базам данных.

Вопросы обеспечения надежности особенно важны в приложениях уровня предприятия для работы с критически важными данными. С этой целью *PostgreSQL* позволяет настраивать горячее резервирование, восстановление на заданный момент времени в прошлом, различные виды репликации (синхронную, асинхронную, каскадную).

PostgreSQL позволяет работать по защищенному SSL-соединению и предоставляет большое количество методов аутентификации, включая аутентификацию по паролю, клиентским сертификатам, а также с помощью внешних сервисов (LDAP, RADIUS, PAM, Kerberos).

При управлении пользователями и доступом к объектам БД предоставляются следующие возможности:

– создание и управление пользователями и групповыми ролями;

– разграничение доступа к объектам БД на уровне как отдельных пользователей, так и групп;

– детальное управление доступом на уровне отдельных столбцов и строк;

– поддержка *SELinux* через встроенную функциональность *SE-PostgreSQL* (мандатное управление доступом).

PostgreSQL обеспечивает полную поддержку свойств ACID и обеспечивает эффективную изоляцию транзакций. Для этого в *PostgreSQL* используется механизм многоверсионного управления одновременным доступом (MVCC). Он позволяет обходиться без блокировок во всех случаях, кроме одновременного изменения одной и той же строки данных в нескольких процессах. При этом читающие транзакции никогда не блокируют пишущие транзакции, а пишущие – читающих. Это справедливо и для самого строгого уровня изоляции *serializable*, который, используя инновационную систему *Serializable Snapshot Isolation*, обеспечивает полное отсутствие аномалий сериализации и гарантирует, что при одновременном выполнении транзакций результат будет таким же, как и при последовательном выполнении[13].

Для написания графического интерфейса выбран фреймворк *Kivy*.

Kivy был создан в 2011 году. Данный кросс-платформенный фреймворк *Python* работает на *Windows*, *Mac*, *Linux*, *Android*, *IoS* и *Raspberry Pi*. В дополнение к стандартному вводу через клавиатуру и мышь он поддерживает мультитач. *Kivy* даже поддерживает ускорение GPU своей графики, что во многом является следствием использования *OpenGL ES2*. У проекта есть лицензия MIT, поэтому библиотеку можно использовать бесплатно и вкупе с коммерческим программным обеспечением.

Во время разработки приложения через *Kivy* создается интуитивно понятный интерфейс (Natural user Interface), или NUI. Его главная идея в том, чтобы пользователь мог легко и быстро приспособиться к программному обеспечению без чтения инструкций[14].

Кроме того, *Kivy* позволяет создавать приложения с отзывчивым дизайном (расположение и наличие элементов управления зависит от размера экрана).

Сообщество *Kivy* разработало множество библиотек, таких как *HotReloader*, *KivyMD*, *Awesome KivyMD*, *AsyncKivy* и др. Перечисленные библиотеки позволяют ускорить разработку и сделать внешний вид приложения современным.

При разработке программного продукта очень важно протестировать его функциональность. В качестве фреймворка для тестирования выбран *pytest*.

Фреймворк *pytest* помогает легко писать небольшие тесты и масштабируется для поддержки сложного функционального тестирования приложений и библиотек.

Возможности *pytest*:

– подробный разбор упавших проверок assert;

– автообнаружение тестовых модулей и функций;

– использование модульных фикстур для управления небольшими или параметризованными тестовыми ресурсами;

– запуск тестовых наборов, написанных с использованием *unittest* (включая пробные) и *nose*;

– совместим с Python 3.5+ и PyPy 3;

– у *pytest* есть большой набор (более 315 внешних плагинов) и процветающее сообщество.

Сегодня большое число программ доставляются и запускаются в так называемых контейнерах.

Контейнеризация (виртуализация на уровне ОС) – технология, которая позволяет запускать программное обеспечение в изолированных на уровне операционной системы пространствах. Контейнеры являются наиболее распространенной формой виртуализации на уровне ОС. С помощью контейнеров можно запустить несколько приложений на одном сервере (хостовой машине), изолируя их друг от друга.

Процесс, запущенный в контейнере, выполняется внутри операционной системы хостовой машины, но при этом он изолирован от остальных процессов. Для самого процесса это выглядит так, будто он единственный работает в системе. [15]

Схема устройства контейнеров представлена на рисунке 1.4.

Рисунок 1.4 – Схема устройства контейнеров

Docker – это программная платформа для быстрой разработки, тестирования и развертывания приложений. *Docker* упаковывает ПО в стандартные блоки, которые называются контейнерами.

В рамках диплома технология *Docker* будет использоваться для запуска СУБД *PostgreSQL* и хранения данных, а также для запуска тестов.

При написании программного кода рекомендуется придерживаться определенного формата. В *Python* существует *PEP 8* – руководство по написанию кода. Для того, чтобы форматировать код в соответствии со стандартом, будет использоваться технология *Black*.

Black – это фреймворк, который может автоматически изменять форматирование *python*-файлов.

В качестве системы контроля версий выбрана *Git*.

Система контроля версий (СКВ) – это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определенным старым версиям этих файлов.

1.4 Постановка задачи на дипломное проектирование

Целью написания данной работы является разработка программного комплекса управления расписанием в ГГТУ им. П. О. Сухого, позволяющего формировать расписание в автоматическом режиме.

В результате проведенного аналитического обзора существующих аналогов были сформулированы следующие функциональные требования к проектируемому программному обеспечению:

– строить расписание без «окон» у учебных групп;

– оптимизировать в расписании «окна» преподавателей;

– учитывать требуемый диапазон дней/часов для классов, для преподавателей и для аудиторий;

– учитывать характер работы и пожелания преподавателей;

– оптимально размещать занятия по кабинетам (аудиториям) с учетом особенностей учебных групп, предметов, приоритетов преподавателей и вместимости кабинетов;

– вводить расписание звонков;

– в ручном режиме заполнения расписания иметь возможность просмотра расписания одновременно в двух видах (всего четыре вида: расписание преподавателей, заполнение аудиторий, расписание учебных групп, окно рабочих нагрузок);

– устанавливать время перехода (переезда) между учебными корпусами;

– оптимизировать количество переходов из кабинета в кабинет, и из корпуса в корпус;

– легко соединять любые учебные группы в потоки при проведении любых занятий;

– разделять учебные группы при проведении занятий по иностранному языку, физической культуре (и любым другим предметам) на любое количество подгрупп;

– вводить комбинированные уроки для подгрупп (типа «иностранный/информатика») по любым предметам;

– легко и быстро вводить и корректировать исходные данные;

– автоматически преобразовывать расписания при изменении базы данных;

– быстро вносить любые необходимые корректировки в расписание;

– находить замены временно отсутствующих преподавателей;

– автоматически контролировать расписание, исключая любые «накладки» и противоречия;

– выводить расписания в виде PDF-документов, готовых к экспорту на сайт университета.

Исходными данными к проекту являются:

– база данных университета (нужны не все таблицы, а только те, которые связаны с расписанием);

– наряды на преподавателей;

– ФИО ответственных лиц за составление и внедрение расписания;

– расписание звонков.

Предполагаемыми выходными данными являются:

– PDF-документы с расписанием.

В силу того, что приложение оперирует открытыми данными (которые можно найти на официальном сайте университета ГГТУ им. П. О. Сухого), оно может быть установлено на любую машину. С точки зрения безопасности, важными данными являются имя пользователя и пароль в PostgreSQL. Они могут быть получены лишь при непосредственном доступе к машине, на которой установлена программа.

Системные требования для работы приложения: Windows 7/10, 4 Гб ОЗУ, 1000 Мб на жестком диске, разрешение экрана 1920 × 1080.

Разрабатываемое приложение должно вести журнал действий пользователя и обрабатывать любые пользовательские ошибки. Выводить соответствующие совершенным действиям уведомления и, если требуется, подсказки. Допускается длительная работа (более одного астрономического часа) программы при составлении расписания в силу сложности процесса и отсутствии универсального алгоритма.

2 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Функциональное моделирование

2.1.1 В настоящее время для разработки функциональных моделей программного обеспечения используется унифицированный язык моделирования.

Унифицированный язык моделирования (UML) – это семейство графических нотаций, в основе которого лежит единственная метамодель. Он помогает в описании и проектировании программных систем, в особенности систем, построенных с использованием объектно-ориентированных (ОО) технологий[6].

UML – точный язык, поэтому можно было бы ожидать, что он основан на предписывающих правилах. Но UML часто рассматривают как программный эквивалент чертежей из других инженерных дисциплин, а эти чертежи основаны не на предписывающих нотациях. Никакой комитет не говорит, какие символы являются законными в строительной технической документации; эти нотации были приняты по соглашению, как и в естественном языке[6].

Основным компонентом языка UML является диаграмма.

Диаграмма (diagram) – графическое представление совокупности элементов модели в форме связного графа, вершинам и ребрам (дугам) которого приписывается определенная семантика.

В нотации языка UML определены следующие виды канонических диаграмм:

– вариантов использования (use case diagram);
– классов (class diagram);
– кооперации (collaboration diagram);
– последовательности (sequence diagram);
– состояний (statechart diagram);
– деятельности (activity diagram);
– компонентов (component diagram);
– развертывания (deployment diagram).

Диаграммы представляют статическую структуру приложения (диаграммы вариантов использования, классов и др.), поведенческие аспекты разрабатываемой программной системы (диаграмма деятельности), физические аспекты функционирования системы (диаграммы реализации).

Еще одним компонентом языка является сущность. К сущностям UML относятся структурные сущности (классы, интерфейсы, прецеденты), поведенческие сущности (автоматы), групповые сущности (пакеты), аннатационные сущности (примечания) [7].

2.1.2 Разрабатываемый программный комплекс предназначен для управления расписанием университета и предоставляет диспетчерам возможность перехода от ручного режима составления расписания к автоматическому.

В составлении расписания выделяют следующие роли: диспетчер, методист кафедры, методист деканата, преподаватель.

Подразумевается, что база данных университета содержит все необходимые сведения о преподавателях, группах, дисциплинах и аудиториях в университете, т.к. именно база данных является источником исходных данных для автоматического формирования расписания.

Для того, чтобы внедрить программный комплекс, необходимо, чтобы методист кафедры имел частичный доступ к базе данных университета и имел права на добавление/удаление/редактирование записей таблицы, хранящей сведения о нарядах на преподавателей. Пожелания преподавателей также добавляются в виде записей в эту таблицу и будут учтены в дальнейшем при составлении расписания.

Функциональная модель «Составление расписания» может быть описана при помощи диаграммы вариантов использования, которая представлена на рисунке 2.1.

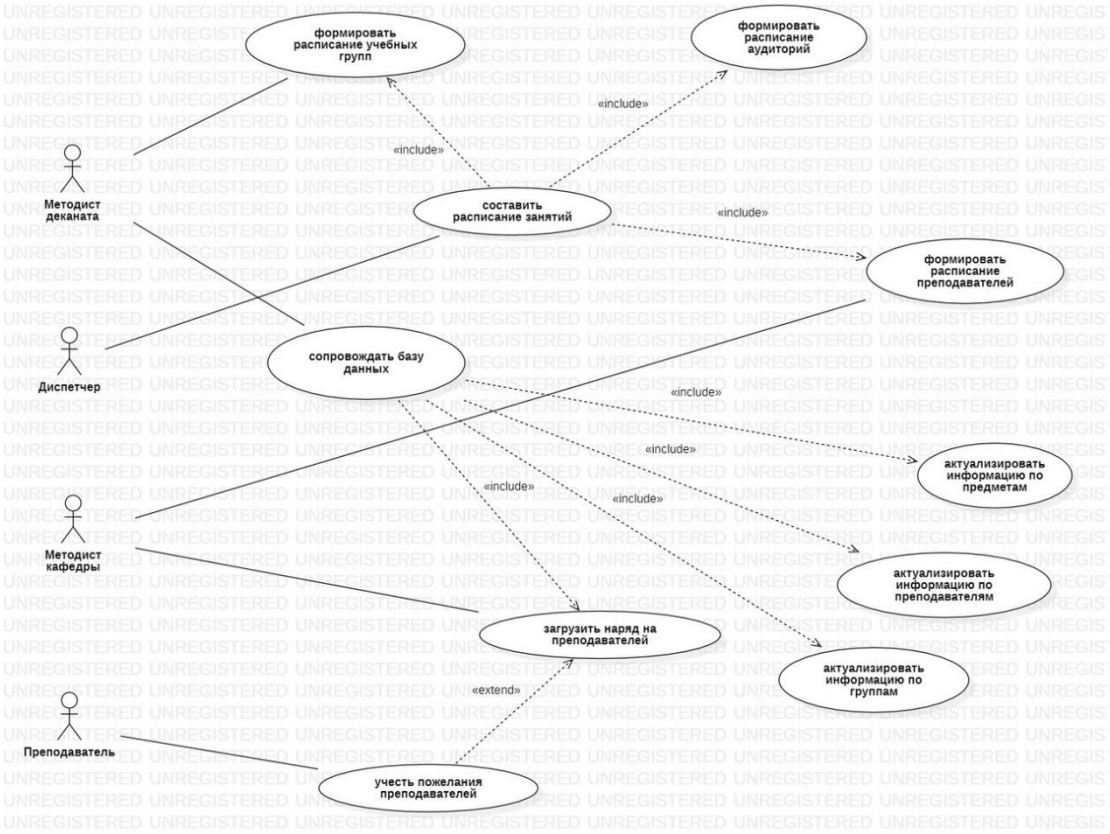

Рисунок 2.1 – Диаграмма вариантов использования функциональной модели «Составление расписания»

На рисунке 2.2 представлена диаграмма IDEF0 функциональной модели «Составление расписания».

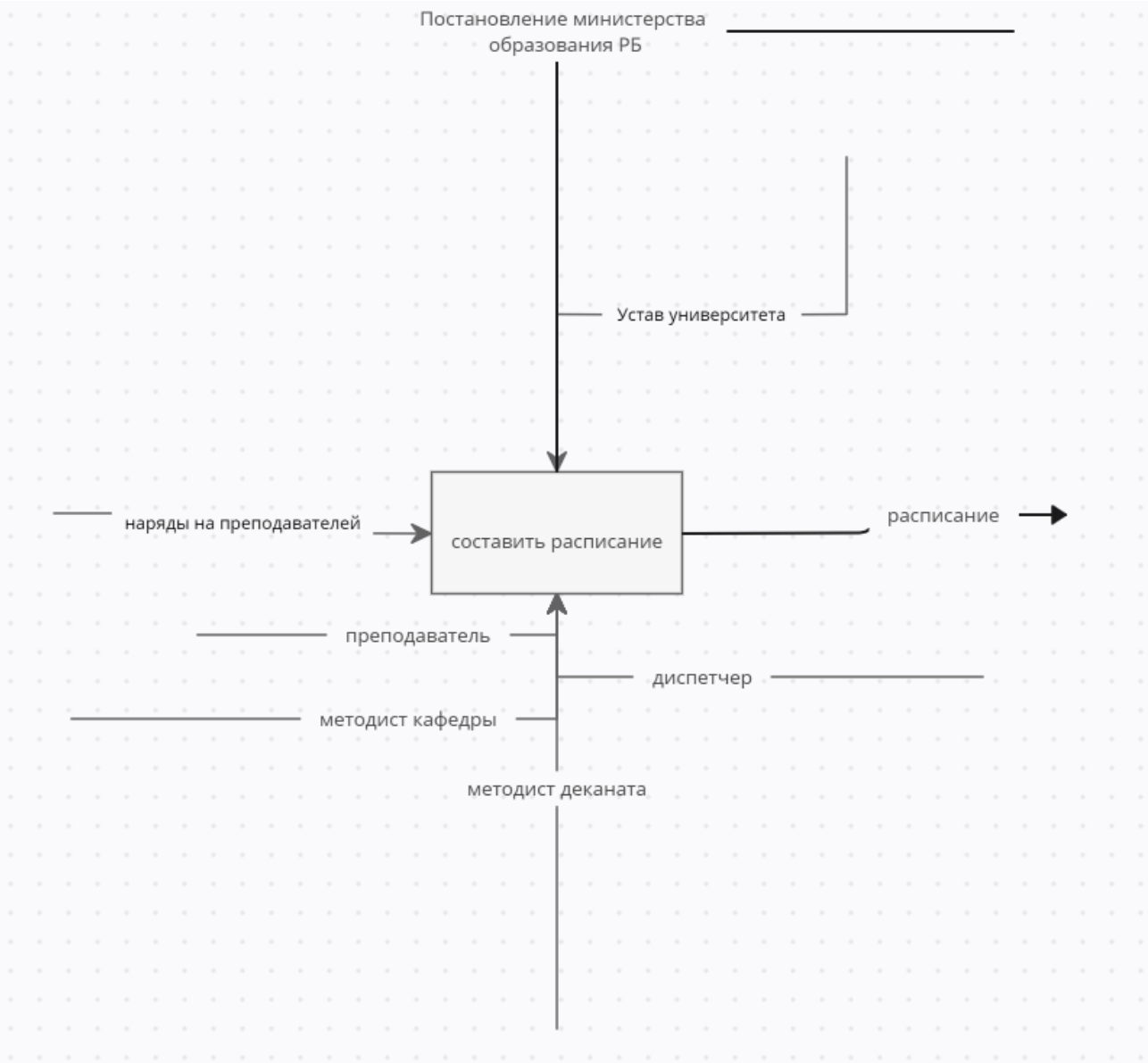

Рисунок 2.2 – Диаграмма IDEF0 функциональной модели «Составление расписания»

2.2 Разработка информационной модели

В результате анализа предметной области, а также ее функциональной модели, были выделены следующие классы, представляющие систему в целом:

– факультет;

– кафедра;

– учебная группа;

– преподаватель;
– предмет;
– тип предмета;
– расписание;
– аудитория;
– рабочая нагрузка;
– запись расписания;
– аудитория занятия.

Класс «факультет» описывает факультеты университета и содержит следующие поля:

– наименование;
– глава.

Класс «кафедра» описывает кафедры университета и содержит следующие поля:

– наименование;
– код факультета;
– глава.

Класс «учебная группа» описывает учебные группы и содержит следующие поля:

– наименование;
– число студентов;
– код факультета.

Класс «преподаватель» описывает преподавателей и содержит следующие поля:

– ФИО;
– ученая степень;
– код кафедры.

Класс «предмет» описывает учебные дисциплины и содержит следующие поля:

– наименование.

Класс «тип предмета» описывает типы учебных дисциплин и содержит следующие поля:

– наименование.

Класс «расписание» описывает расписания и содержит следующие поля:

– год;
– семестр.

Класс «аудитория» описывает аудитории и содержит следующие поля:

– номер;
– число посадочных мест;
– код кафедры.

Класс «рабочая нагрузка» описывает аудитории и содержит следующие поля:

– код группы;
– код преподавателя;
– код предмета;
– код типа предмета;
– год;
– семестр;
– количество часов в неделю.

Класс «запись расписания» описывает аудитории и содержит следующие поля:

– код группы;
– код преподавателя;
– код предмета;
– код типа предмета;
– код аудитории;
– код расписания;
– номер пары;
– день недели;
– тип недели;
– подгруппа;
– свободен преподаватель.

2.3 Проектирование базы данных

Для проектирования базы данных существуют различные методики, различные последовательности шагов или этапов. В целом, можно выделить следующие этапы:

– выделение сущностей и их атрибутов, которые будут храниться в базе данных, и формирование по ним таблиц;

– определение уникальных идентификаторов (первичных ключей) объектов, которые хранятся в строках таблицы;

– определение отношений между таблицами с помощью внешних ключей;

– нормализация базы данных.

В настоящее время для упрощения работы с базами данных используются ORM-библиотеки. Они позволяют взаимодействовать с базами данных с помощью объектно-ориентированного кода, не используя SQL-запросы.

Одной из самых популярных ORM-библиотек для Python сегодня является *SQLAlchemy*. У нее есть особенность – код приложения будет оставаться тем же

вне зависимости от используемой базы данных. Это позволяет с легкостью мигрировать с одной базы данных на другую, не переписывая код приложения.

С выходом последней версии *SQLAlchemy* (версия 2.0) появилась возможность создания универсальных классов, позволяющих одновременно описывать сущности, их атрибуты, связи и ограничения, а также использовать экземпляры данных классов в качестве объектов-значений.

На рисунке 2.3 приведена логическая модель спроектированной базы данных для программного комплекса.

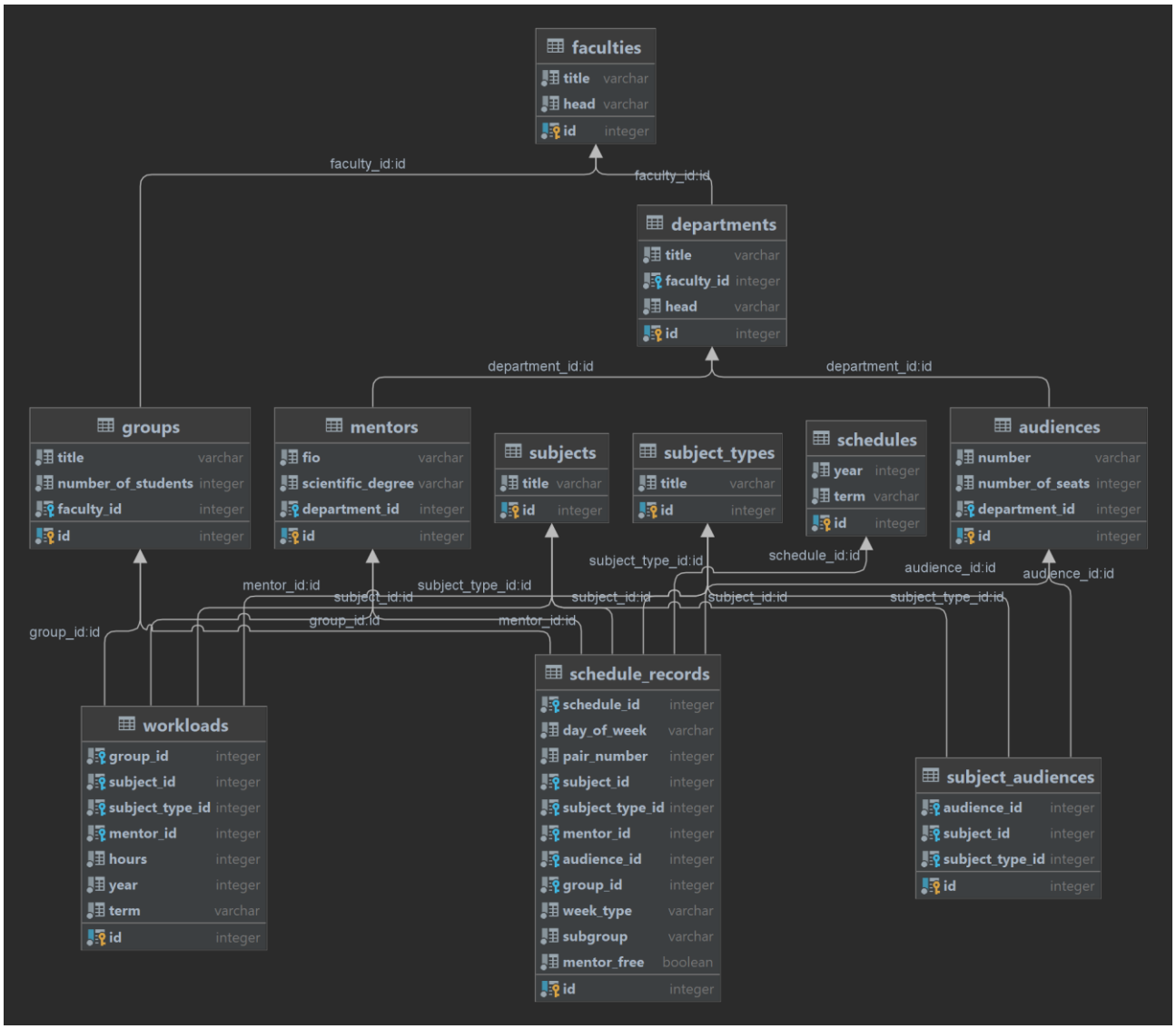

Рисунок 2.3 – Логическая модель спроектированной базы данных для программного комплекса

Сущность «schedules» представляет собой информацию о имеющихся расписаниях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.1.

Таблица 2.1 – Таблица «schedules»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
year	varchar	Да	Год
term	varchar	Да	Семестр

Сущность «audiences» представляет собой информацию о имеющихся аудиториях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.2.

Таблица 2.2 – Таблица «audiences»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
number	varchar	Да	Номер аудитории
number_of_seats	integer	Да	Число посадочных мест
department_id	integer	Нет	Внешний ключ (кафедра)

Сущность «departments» представляет собой информацию о имеющихся кафедрах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.3.

Таблица 2.3 – Таблица «departments»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
title	varchar	Да	Название
faculty_id	integer	Нет	Внешний ключ (факультет)
head	varchar	Да	ФИО зав. кафедрой

Сущность «faculty» представляет собой информацию о имеющихся факультетах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.4.

Таблица 2.4 – Таблица «faculty»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
title	varchar	Да	Название
head	integer	Да	ФИО декана

Сущность «groups» представляет собой информацию о имеющихся учебных группах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.5.

Таблица 2.5 – Таблица «groups»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
title	varchar	Да	Название
number_of_students	integer	Да	Число студентов
faculty_id	integer	Нет	Внешний ключ (факультет)

Сущность «mentors» представляет собой информацию о имеющихся преподавателях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.6.

Таблица 2.6 – Таблица «mentors»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
fio	varchar	Да	ФИО
scientific_degree	varchar	Да	Научная степень
department_id	integer	Нет	Внешний ключ (кафедра)

Сущность «schedule_records» представляет собой информацию о имеющихся записях расписания. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.7.

Таблица 2.7 – Таблица «schedule_records»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
schedule_id	integer	Нет	Внешний ключ (расписание)
day_of_week	varchar	Нет	День недели
pair_number	integer	Да	Номер пары
subject_id	integer	Нет	Внешний ключ (предмет)
subject_type_id	integer	Нет	Внешний ключ (тип предмета)
mentor_id	integer	Нет	Внешний ключ (преподаватель)
audience_id	integer	Нет	Внешний ключ (аудитория)
group_id	integer	Нет	Внешний ключ (группа)
week_type	varchar	Да	Тип недели
subgroup	varchar	Да	Подгруппа
mentor_free	bool	Да	Если False, то означает невозможность проведения занятия преподавателем

Сущность «subject» представляет собой информацию о имеющихся предметах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.8.

Таблица 2.8 – Таблица «subject»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	Integer	Нет	Первичный ключ
title	varchar	Да	Название

Сущность «subject_audiences» представляет собой информацию о возможности проведения занятий в определенных аудиториях. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.9.

Таблица 2.9 – Таблица «subject_audiences»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
subject_id	integer	Нет	Внешний ключ (предмет)
subject_type_id	integer	Нет	Внешний ключ (тип предмета)
audience_id	integer	Нет	Внешний ключ (аудитория)

Сущность «subject_types» представляет собой информацию о имеющихся предметах. Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.10.

Таблица 2.10 – Таблица «subject_types»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	Integer	Нет	Первичный ключ
title	varchar	Да	Название

Сущность «workloads» представляет собой информацию о имеющихся рабочих нагрузках (берутся из нарядов преподавателей). Описание полей таблицы базы данных, соответствующей данной сущности, приведено в таблице 2.11.

Таблица 2.11 – Таблица «workloads»

Имя столбца	**Тип данных**	**Разрешение NULL**	**Описание**
id	integer	Нет	Первичный ключ
subject_id	integer	Нет	Внешний ключ (предмет)
subject_type_id	integer	Нет	Внешний ключ (тип предмета)
mentor_id	integer	Нет	Внешний ключ (преподаватель)
group_id	integer	Нет	Внешний ключ (группа)
hours	integer	Да	Количество часов в неделю
year	integer	Да	Год
term	varchar	Да	Семестр

ЗАКЛЮЧЕНИЕ

Во время прохождения преддипломной практики был проведен анализ предметной области, а также аналитический обзор существующих аналогов. Определена цель и поставлены задачи к созданию системы управления расписанием в ГГТУ им. П. О. Сухого.

Также в рамках прохождения преддипломной практики был выбран инструментарий, позволяющий упростить как можно больше процессов и реализовать программный комплекс.

Затем было проведено функциональное моделирование, позволяющее определить такие важные параметры, как:

– входные данные в программный комплекс;

– результирующие данные;

– основные функции и возможности разрабатываемого программного комплекса.

На основе функционального моделирования была частично разработана информационная модель (все, за исключением алгоритма автоматического составления расписания).

Разработка информационной модели позволила перейти к проектированию базы данных.

Результат, полученный в рамках преддипломной практики, является отправным пунктом к написанию программы и ее тестированию.

По окончании преддипломной практики можно создать положительный прогноз на основании проделанной работы о положительном завершении дипломного проектирования.

Список использованных источников

1. Аганина, Д. А. Проблемы автоматизированного расписания образовательного процесса / Д. А. Аганина. // Молодой ученый. – 2018. – № 42 (228). – С. 42-43.
2. Расписание : Свободная энциклопедия. – Электрон. данные. – Режим доступа: https://ru.wikipedia.org/wiki/Расписание#cite_ref-6. Дата доступа: 16.04.2023.
3. Теория расписаний и вычислительные машины / под ред. Э. Г. Коффмана. – М.: Наука, 1984.
4. Конвей, Р. В. Теория расписаний / Р. В. Конвей, В. Л. Максвелл, Л. В. Миллер – М.: Главная редакция физико-математической литературы изд-ва "Наука", 1975.
5. Танаев В.С. Введение в теорию расписаний / В.С. Танаев, В.В. Шкурба – М.: Главная редакция физико-математической литературы изд-ва "Наука", 1975.
6. Фаулер, М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с.
7. Трохова, Т. А. Функциональное моделирование программных систем в UML : методические указания по курсу "Технологии проектирования программного обеспечения информационных систем" для слушателей специальности 1-40 01 73 "Программное обеспечение информационных систем" / Т. А. Трохова. – Гомель : ГГТУ им. П. О. Сухого, 2012. – 34 с.
8. Кузьмичев, А. Б. О подходе к автоматизации составления расписания в учебном заведении // Техника машиностроения : журнал. – 2014. – № 3. – С. 23-26.
9. Авторасписание. – Электрон. данные. – Режим доступа: https://www.mmis.ru/programs/avtor. Дата доступа: 16.04.2023.
10. Электронное расписание. – Электрон. данные. – Режим доступа: http://it-institut.ru/#top_content. Дата доступа: 16.04.2023.
11. Альтхофф, Кори. Сам себе программист. Как научиться программировать и устроиться в Ebay? – Пер. с англ. М. А. Райтмана. – М: Эксмо, 2018. – 208 с.
12. Возможности – PyCharm. Электрон. данные. – Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/features/. Дата доступа: 16.04.2023.
13. Лузанов, П. PostgreSQL для начинающих / П. Лузанов, Е. Рогов, И. Лёвшин. – 3-е изд. – Минск : ДМК, 2017. – 118 с.
14. Создание приложений для Android, iOS, macOS и Windows на Python. Электрон. данные. – Режим доступа: https://python-scripts.com/kivy-android-ios-exe. Дата доступа: 16.04.2023.
15. Основы контейнеризации (обзор Docker и Podman) : Хабр. Электрон. данные. – Режим доступа: https://habr.com/ru/articles/659049/. Дата доступа: 16.04.2023.